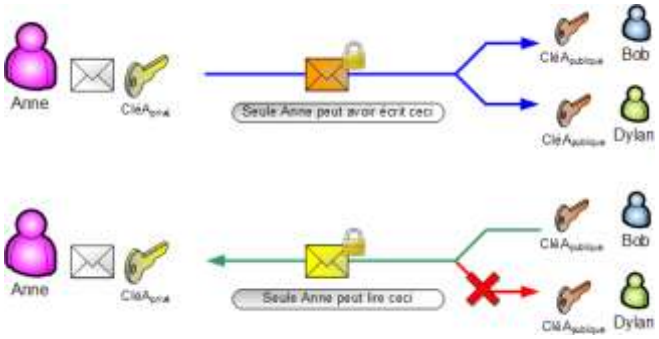
TP8 – CRYPTAGE RSA

Le chiffrement RSA est un procédé de cryptage de données couramment utilisé pour les transmissions par internet et le commerce électronique.

Son nom est composé des initiales des trois informaticiens qui l'ont défini en 1977 : Ronald Rivest, Adi Shamir et Leonard Adleman.



Le principe est le suivant :

- *Les données sont numérisées, c'est-à-dire converties en entiers naturels*

Tous ces entiers naturels sont supposés inférieurs à un entier **n** fixé, appelé **module de chiffrement**.

- *Un entier est crypté en appliquant la fonction* $\boldsymbol{F_{n,c}}$ *qui à un entier naturel* ***a*** *associe le reste de la division euclidienne de* $\boldsymbol{a^c}$ *par* ***n****, où* ***c*** *est un entier naturel, appelé* ***clef publique****.*
- *Pour décrypter il suffit d'appliquer la fonction* $\boldsymbol{F_{n,d}}$ *c'est-à-dire la même fonction mais en remplaçant* ***c*** *par* ***d****, entier naturel appelé* ***clef privée****.*
- *Ensuite on convertit les entiers au format de départ, par exemple du texte.*

L'utilisateur qui souhaite recevoir des donnés cryptées communique le module de chiffrement et la clef publique ***(n, c)*** à tous les utilisateurs souhaitant lui envoyer des données.

Il ne communique pas la clef privée **d**, c'est elle qui lui permet de décrypter les données.

1. RAPPELS : CODAGE DES CARACTERES

Chaque caractère est codé par un entier, c'est le codage ASCII étendu. Par exemple les lettres minuscules a à z sont codées de 97 à 122.

En python la fonction **ord** donne le code ASCII d'un caractère, tandis que la fonction **chr** donne le caractère codé par l'argument.

Testez :

```
>>> ord('a')
>>> chr(122)
```

Q1. Obtenir les caractères codés par les entiers de 32 à 127.

Q2. Écrire une fonction ***Numerisation*** *qui reçoit une chaine de caractères et qui renvoie la liste des codes ASCII de ses lettres.*

Par exemple **Numerisation("Test")** doit renvoyer [84,101,115,116].

Q3. Écrire la fonction ***ConversionStr*** *qui reçoit une liste d'entiers et qui renvoie la chaine de caractère qu'elle code. Tester avec l'exemple précédent.*

2. CRYPTAGE

Q4. Écrire une fonction ***F(a,n,c)*** *qui à l'entier* ***a*** *associe le reste de la division euclidienne de* a^c *par* ***n***.

Tester si F(37,199,25) renvoie bien 141.

Q5. Écrire une fonction ***RSA(L,n,c)*** *qui reçoit une liste d'entiers* ***L*** *et qui renvoie la liste des éléments* ***F(a,n,c)*** *où* ***a*** *parcourt la liste* ***L***.

Tester si RSA([0,1,2,3,4],199,85) renvoie [0,1,196,149,9]

Q6. Écrire une fonction ***Cryptage(S,n,c)*** *qui reçoit une chaine de caractère* ***S*** *et qui renvoie le cryptage RSA de* ***S*** *avec* ***n*** *pour module de chiffrement et* ***c*** *pour clef.*

Tester si Cryptage("Test",199,85), renvoie "KS|o".

Q7. Que renvoie l'instruction ***Cryptage("KS|o",199,7)*** *? Que peut-on en conclure ?*

3. APPLICATION

Pour cette partie on fixe **n = 259**.

La suite de l'énoncé a été cryptée dans le but que seuls vous puissiez la lire.

La clef privée dont vous disposez est **d = 85.**

Décrypter le texte et continuer le TP (Q8 à Q12) !

Attention : copier/coller le texte depuis le fichier **Partie3.txt** en tant que chaine de caractères dans une variable, par exemple :

```
Partie3= """ coller le texte ici """
```

Q13. Quelle clef ai-je utilisé pour crypter mon texte ? Là encore deux valeurs sont possibles.

Voici quelques éléments qui n'ont pas été exposés jusqu'ici :

L'entier ***n*** doit être le produit de deux nombres premiers distincts ***p*** et ***q*** : ***n = pq***.

La valeur de ***n*** est publique, mais en pratique ***p*** et ***q*** sont très grands (supérieurs à 10300). On ne connaît pour l'instant aucun algorithme permettant à un ordinateur actuel , même puissant, de déterminer ***p*** et ***q*** à partir de ***n***.

Si ***p*** et ***q*** sont connus, alors on peut décrypter le texte. En effet la valeur de la clef publique ***c*** est connue, il reste à déterminer celle de la clef privée ***d***. Ces deux clefs sont reliées par la relation suivante :

Le reste de la division euclidienne du produit ***cd*** par ***(p -1)(q - 1)*** est égal à 1.

(On note ***φ(n) = (p - 1)(q - 1),*** c'est ***l'indicateur d'Euler*** de ***n***.)

Par exemple pour ***n = 259***, on a ***259 = 7*37***, donc ***p = 7*** et ***q = 37*** quitte à les inverser. Alors ***(p - 1)(q - 1) = 6 *36 = 216,*** ce qui explique la valeur 216 donnée par l'énoncé précédemment !

Cette détermination de ***d*** connaissant ***c*** est possible en temps raisonnable par les ordinateurs actuels.

Q14. On pose p = 6 311 et q =9 743, puis c = 38 397 531.

Écrire un programme déterminant la clef privée d associée en testant toutes les possibilités, i.e., pour ***d*** *allant de* ***1*** *à* ***(p - 1)(q - 1).***
Donner le résultat.
Répondre à la même question avec ***p = 68 351*** *et* ***q = 92 893****, puis* ***c = 4 857 201 829****.*
Répondre à la même question avec ***p = 652 429*** *et* ***q = 936 527****, puis* ***c = 453 710 465 897****.*

Ne pas attendre la fin de l'opération !

On constate bien qu'il est nécessaire d'avoir un algorithme plus efficace.

Cet algorithme existe, c'est ***l'algorithme d'Euclide étendu***. Voici son fonctionnement :

Les variables ***r, s, u, v*** reçoivent les valeurs respectives ***c, φ(n), 1, 0***.

Tant que ***s*** est strictement supérieur à **0**, calculer ***q*** le quotient de la division euclidienne de ***r*** par *s*, puis remplacer

- ***r, s*** par ***s, r - qs***
- ***u, v*** par ***v, u - qv***.

À l'issue de la boucle, ***d = φ(n) + u*** contient la valeur souhaitée.

Q15. Programmer cet algorithme.

Q16. On pose p = 652 429 et q = 936 527, puis c = 453 710 465 897.

*Déterminer la clef privée **d** à l'aide de l'algorithme défini ci-dessus.*
Répondre à la même question avec p = 688 257 281 et q = 954 592 879, puis c = 468 972 057 559 384 321.

Table ASCII :

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□